AP Interoperability Issues In STEP

L. J. McKee
October 7, 1994

Table of Contents

1 Background

The purpose for this paper is to explore the ability of STEP, in its current form to enable data sharing. This one facet of this process is colloquially know as AP interoperability and involves an implementation of one AP being able to access and use data defined in an implementation of another AP. It further involves the desire to create implementations which are capable of storing and sharing the data for many APs concurrently.

ISO 10303, also know as STEP, is described in the introduction of each volume or part as follows:

> "ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving."

These statements lead the reader to believe that STEP will facilitate the sharing of product data between companies and also between disciplines within companies. This ability is unique to STEP, at present, as a data standard. This ability would allow STEP to serve as the underlying data infrastructure for concurrent engineering in a virtual enterprise whether national or international.

This paper will look at where the current STEP data model set (resource and APs) seem to fall short of this lofty goal. The "look" taken in this paper will be from both the "vanilla" STEP point of view and the implementation point of view. It will further explore ways in which this condition could be corrected. One of the fundamental assumptions on which this paper is based is that AP global rules (all global rules?) should be consitently implemented across all proposed STEP implementation forms (physical files, working forms, databases, and knowledge bases).

It is not the intent of this paper to assess blame for the indicated areas or possible problems. This paper is solely intended to put forth the views of the author on how well STEP currently addresses the aforementioned goal.

2 Current Status

There have been many sessions in many organizations in which STEP and its ability to share data have been discussed. The status stated here is the current state of affairs as discussed and documented in the efforts of the PDES, Inc. industry consortium.

PDES, Inc. had initial discussions on the topic of interoperability at the spring offsite in March of 1994 in Myrtle Beach. These discussions dealt with areas which had been identified through the Advanced Weapon System pilot project where the concurrent storage of data for AP 203

(Configuration Controlled Design) and AP 202 (Associative Draughting) seemed not to be possible.

These initial discussions were followed by an interoperability summit where the majority of PDES, Inc. AP teams were represented. The goal of this session was to explore a methodology through which interoperation and data sharing was possible with minimal impact on the data models in STEP release 1.0.

The result of this session was that it is unreasonable to assume that an implementation of one AP would be capable of using data from an implementation of another AP without having access to the AP schema in which the data was defined. This conclusion was extrapolated upon and the conclusion reached at that time was that data sharing to the maximum extent possible was feasible if an AP implementation is given data from the implementation of another AP and the schema from that AP.

The remainder of this paper will deal with potential pitfalls in this methodology from the author's view. In order to do this adequately, the reader must have a better understanding of the "problem" which led to the methodology and a better understanding of the methodology itself.

2.1 The Problem

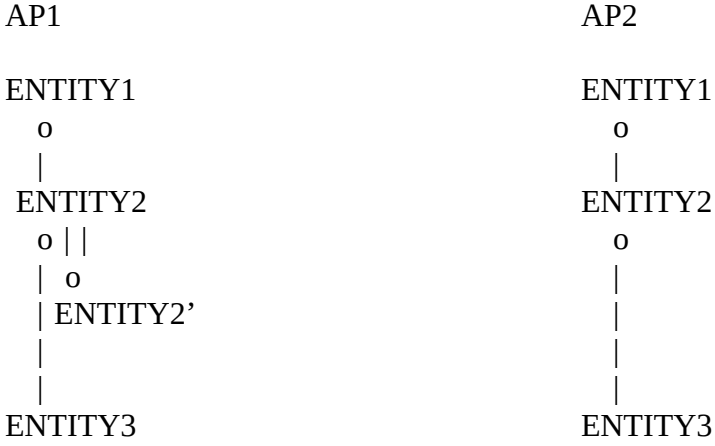
The ability for STEP to facilitate both data exchange and sharing is predicated on the fact that all implementations of STEP must be based on the STEP APs and that these APs are all derived from a common set of resource models. In the formation of an AP, an application process model is used to derive and scope an application data model which is mapped into the STEP resource models with entity specializations and population constraints created to satisfy the intent of the application process model. In other words, an application activity model (AAM) is used to derive and scope an application reference model (ARM) which is mapped into the STEP resources (40 and 100 series models) with specializations (AP specific entity SUBTYPEs) and population constraints (EXPRESS rules).

Each STEP AP is created to exchange and store the data for **one** application area. The AP author (group or single person) is only interested in the creation of a data model to satisfy the application in which he/she/they have interest. There is no requirement that the AP author investigate the relationship of the application he/she/they are modeling to any other application or AP. There is also, currently, no group in the STEP sponsoring body (ISO TC 184/SC 4) which is monitoring or evaluating these interfaces.

The STEP development environment is a ripe breeding ground for disconnects and general interface problems amongst APs due to the aforementioned lack of monitoring. These problems surface when AP models are brought together and data is applied to them. The problems stem from the lack of attention by any group to AP interfaces. The symptoms of these problems vary widely and are dependent on the applications being brought together. In a generalized form, the symptoms show themselves as data population and exchange problems caused by the AP entity

specializations and constraints.

This can be easily illustrated. Assume we have two APs with the following schemas based entirely of STEP resource model entities:



AP1 has an SUBTYPE of ENTITY2 (ENTITY2') which is not utilized in AP2. If an implementation of AP2 uses just the schema provided in AP2, it has no knowledge that ENTITY2' exists. If an exchange via physical file were attempted from AP1 to AP2, and AP1 requires the use of ENTITY2', the file would contain the following:

#1=ENTITY1(...);
#2=ENTITY2'(...,#1,...);
#3=ENTITY3(...#2...);

When an implementation of AP2 attempts to read this file and store the data, it runs across ENTITY2' and declares it an illegal entity. The connection path from ENTITY1 to ENTITY3 is lost, and the data in ENTITY2/2' is lost. It is quite possible that the entire exchange could be lost as well.

The problem goes beyond just what is brought out in this simple example. AP2 may require in rules entity relationships which are not required (but contained) in AP1. AP2 may also more tightly constrain the population of attributes in the common entities with AP1.

2.2 The Proposed Solution

What is to be done about these situations? Enter the proposed PDES, Inc. implementation methodology as a solution. The implementation methodology states that in order to get the best exchange (or sharing) possible, implementations must look at not only the data from another AP, but also its schema. The implementation should preserve as much data as possible.

Let’s see how this helps in the simple example. Since the implementation of AP2 now can have access to the schema for AP1, it can decipher that ENTITY2’ is a SUBTYPE of ENTITY2. This piece of information now makes it possible for the implementation of AP2 to preserve the data that is in the attributes of ENTITY2 and the connection from ENTITY1 to ENTITY3. The only data which is lost is that unique to ENTITY2’.

This is clearly a better situation. It will work for resource part SUBTYPES which are shared and for AP specific subtypes which are not shared. The implementation methodology actually does more than this. It further states that when one AP reads data from another it should not enforce its rules on the data coming from the other AP. The enforcement of AP rules is done when the data is tagged as being conformant to the reading AP’s schema by attaching an **application_ protocol_definition** entity with the reading AP’s schema name.

This second stage of the implementation methodology is to eliminate problems in data sharing caused by rule conflicts. Prior to adding the **application_protocol_definition** entity with the reading AP’s schema name, the receiver of the data must work with the data to ensure that all rule violations are corrected. This is done by enhancing the data so that it does not violate any of the rules in the reading AP.

The proposed implementation methodology does make things better. The remainder of this paper will look at potential problems in employing this methodology and discuss solutions to those problems for both the short and the long term.

3 Case Study Using APs 202 and 203

This section and its subsections will look at how well the currently proposed PDES, Inc. implementation methodology would perform using APs 203 and 202 as a discussion point.

3.1 Overview

Upon reading the scope statements in APs 203 and 202, it is obvious there is an overlap in that both APs deal with three dimensional shape. This is reflected in the AICs in AP 202 and the plans for incorporation of AICs into AP 203. The two APs share the data described in the 5 subtypes of **shape_representation** that exist in AP 203.

If we look further we find that AP 202 states "...representing drawings for the purpose of exchange, suitable for mechanical engineering and architecture, engineering, and construction applications". AP 203 states it deals with products "that are mechanical parts and assemblies". This would indicate that mechanical products is and area of overlap or commonality. AP 202 also states that it is capable of representing drawings for "any phase of design". AP 203 states that it deals with "Product definition data and configuration control data pertaining to the design phase of a product’s development". This would suggest that both APs cover data on mechanical products during the design cycle. AP 202 goes on to state that it supports "the administrative data identifying the product versions being documented by the drawing". AP 203 states it supports

"The change of a design and data related to the documentation of the change process". Given the prior statements one can extrapolate that AP 202 will support the definition of drawing data for mechanical products in the design phase of development and will identify which version of the product is represented in the drawing. On further extrapolation, we can state that AP 202 should be able to support the definition of drawing data for versions of products represented as 3D models in AP 203.

For the purposes of this paper, this case study will explore whether AP 202 is capable of utilizing shapes stored in AP 203 and vise versa. This will be explored using the methodology proposed at the PDES, Inc. AP interoperability summit that data sharing is possible if an AP implementation is given data from the implementation of another AP and the schema from that AP.

3.2 Physical Files

STEP physical files are essentially ASCII representations of the STEP EXPRESS data structure which contain instance data. These files when produced by a conforming AP implementation also inherently contain the EXPRESS rules and informal propositions from the AP.

In the case of APs 203 and 202, most implementing organizations will use AP 203 for exchanging and sharing information on configuration management (CM) and nominal design shapes created during the conceptual design phase. They will use AP 202 for exchanging and sharing the draughting representation created in the detail design and later phases of the product life cycle. Since the draughting representation is typically a more detailed view of the conceptual design shape, there is a typical desire for AP 202 implementations to be able to utilize shape data from AP 203 implementations.

In the current industrial environment, the converse must also be explored. Since both AP 203 and 202 have 3D shape in common and 3D shape is created (most commonly) by CAD systems, it is entirely plausible that an organization with an AP 203 implementation will attempt to utilized shape data from an AP 202 implementation.

In current implementations, the focus is on pushing the 3D shape in one direction or another. The typical form is that shown in Figure 1 where 203 shape data is brought into a CAD system and used as the basis for the creation of the drawing for the shape.

The type of exchange shown in Figure 1 is possible using 203 and 202 even if the current PDES, Inc. implementation methodology were not employed. This is true since all the interfacing occurs in the CAD system. There is a large amount of data loss since the configuration management data is not used in the creation of 202 file.

The data loss can be limited by employing the PDES, Inc. implementation methodology. The use of this methodology would allow a great deal of useful information to be made available to the 202 implementation since the product identification information could be preserved. This

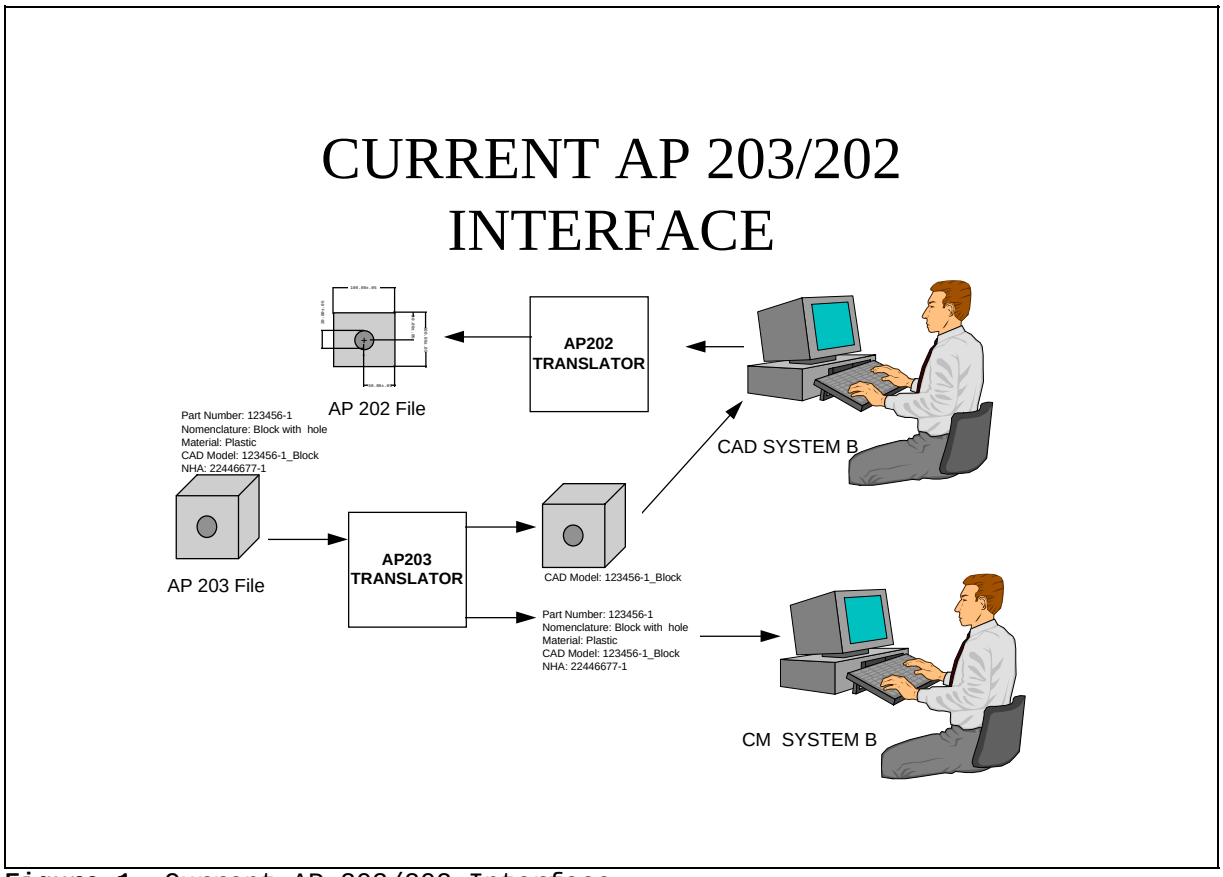


Figure 1 Current AP 203/202 Interface

preservation is possible since the 203 and 202 APs have a "problem" very similar to the earlier example.

It looks like this:

203

```
PRODUCT
   o
   |
 PRODUCT_DEFINITION_FORMATION
   o | |
   |  o
   | PRODUCT_DEFINITION_FORMATION_
   | WITH_SPECIFIED_SOURCE
   |
PRODUCT_DEFINITION
```

202

```
PRODUCT
   o
   |
PRODUCT_DEFINITION_FORMATION
   o
   |
   |
   |
   |
PRODUCT_DEFINITION
```

Without using the PDES, Inc. implementation methodology, the only "sharing" possible between APs 203 and 202 is that of AIC propagation where the common shape AICs can be pushed back and forth. The connection between **product** and shape is lost due to the SUBTYPE "problem". Employing the methodology eliminates this "problem". It also allows the 202 implementation to have access to all the data in 203 which resides in resource model constructs common to the two APs.

This would include all the specification references available in the 203 file via **document** entities which typically appear as notes on the face of the drawing. The use of this information by a 202 implementation would be tricky since the connection entities which are AP 203 specific SUBTYPEs are not resource model entities and not common. This is a little concerning, but, with a human in the loop, could be easily straightened out.

The above does bring up an interesting situation. Let’s now explore where this situation can become a liability to the user. It also becomes a big political liability for STEP. Figure 2 shows short-cut EXPRESS-G notation for pieces of APs 203 and 202. The constructs are very similar as are the usages. If we assume that an exchange is done from AP 202 to AP 203 and that all that is exchanged is geometry, there does not seem to be a problem. The "problem" is that geometry in a **draughting_model** may be purely annotational but three dimensional.

When an implementation of AP 203 looks at this information using the PDES, Inc. implementation methodology, the **draughting_representation** becomes a **representation**. This can cause a **big** problem if the annotation geometry appears to be shape geometry when viewed in a CAD system. The user will simply change the **representation** to either **shape_aspect** or **shape_representation** and go on. What could be even worse would be if the AP 203 processor did this automatically. In this case, there is no way for the user or the processor to determine what should be done. The data is ambiguous. There must be clear guidance provided or mistakes are guaranteed.

This section deals with physical files and their processors. Since physical files are typically processed in a batch environment, the processor can list out warnings and errors for each of the above situations. The processor can also mark extrapolated data as suspect. The processor can not keep people from making mistakes.

The problem presented in the last part of this section needs to be addressed. If left to translator vendors and users, the reputation of STEP will be put on the line. One thing that needs to be looked at is the overuse of resource model entities for dissimilar roles. There may be a need to expand the resource model set rather than reusing constructs in some instances. This is really the only way to increase clarity without sacrificing data sharing.

3.3 Databases

The prior section dealt with physical files. By STEP rules, physical files can only contain the data for one AP per data section. This eliminates the possibility of the data for two APs in the

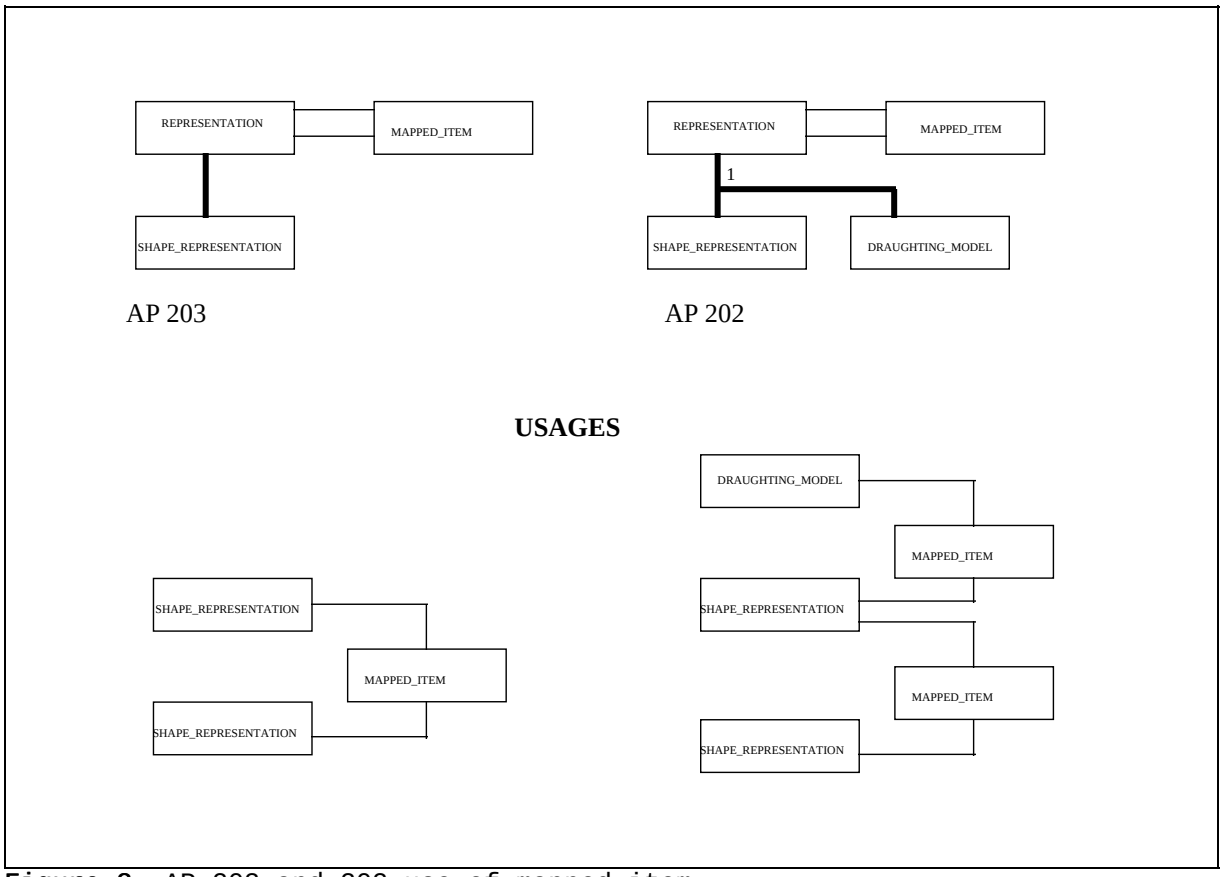


Figure 2 AP 203 and 202 use of mapped_item

same data space. In the database or level 3 implementation world, there is no current guidance and a restriction of that type would cause a vendor and user insurrection.

In the database implementation form, the author will assume that all STEP models could be poured into one database with identical constructs not being replicated. The multi-AP database will be assumed to be the set which is the union of the set of entities in both APs. The PDES, Inc. implementation methodology needs to be re-interpreted for this environment since APs can be combined in a single data store. The author will assume that the database implementation provides SUPERTYPE visibility of SUBTYPE data. (This means that when a SUBTYPE is populated the SUPERTYPE is also populated.)

If the above is the situation, the "problem" becomes that of rules. The current PDES, Inc. implementation methodology assumes that no two APs rules are "on" at the same time. This may not be a problem in a relational environment where the majority of rules are in application code and not part of the data store. In an other environments, this could be a problem.

Another "problem" will occur if the current EXPRESS global rule code is used to generate application code. This will occur in any environment. The reason for this is that most AP rules are global rules. These rules are not in any way tied to anything. If the rules from two APs are

combined as the entities would be in a composite data structure, all hell breaks lose. As an example, consider the following two rules:

```
RULE restrict_document_type FOR (document_type);
WHERE
  WR1: SIZEOF (QUERY (dt <* document_type |
       NOT (dt.product_data_type IN [’material_specification’,
       ’process_specification’, ’design_specification’,
       ’surface_finish_specification’, ’cad_filename’, ’drawing’]))) = 0;
END_RULE;

RULE restrict_document_type FOR (document_type);
WHERE
  WR1: SIZEOF (QUERY (dt <* document_type |
       NOT (dt.product_data_type = ’draughting specification’
       )))=0;
END_RULE;
```

Each of these rules are global rules. If these rules are taken together in the same manner as the entity definitions could be, they conflict. The first rule belongs to AP 203. The second rule belongs to AP 202. The "problem" is that they can not be taken together. The author understands and believes that it was never the intent that they be taken together. This is the crux of the "problem". The rules are built on a physical file paradigm. This would be fine if STEP were only intended to be implemented in a physical file, but that is not the intent.

3.3.1 Rule Anchoring

The point is that global rules used in AP schemas need to be founded in the data rather than the schema. All global rules should be coded to found themselves against a particular instance of **application_protocol_definition**. The author is sure this will bring an outcry from AP owners saying this is too much too ask. If this is true, why is it acceptable to ask the same very thing from implementors?

In either case, this is not an easy task. The point here is that STEP is either implementation independent or it is not. There is no point in saying that STEP entity definitions are implementation independent, but the rules and functions are not.

The rules need to be data anchored. To ease the burden of this task, a tool can be created which would trace the path from the desired entity or entities to the **application_protocol_definition**. Anchoring the rules will correct the situation. Simply saying it is too much work is not the answer. Simply putting it off on the implementors is also not the answer.

Another possible answer could be using the Standard Data Access Interface (SDAI) schema instance for AP rule anchoring. At present, this is not an answer since Part 22 (SDAI) has not

reached DIS status. This method will have to be addressed when the SDAI specification becomes more mature.

3.3.2 AP Insulation

One other thing that might be considered is rule isolation or AP insulation. This method would forbid APs to attach rules to resource model entities. If an AP has a specific requirement which affects one or more resource model entities, the AP would create AP specific SUBTYPEs of the resource entities and attach the rule to these subtypes. Using this paradigm, there can not be any rule conflicts since the rules are only against AP specific subtypes. The one issue here is how would an AP require these SUBTYPEs? This could be done in conformance classes rather than rules. The current conformance class definitions would be enhanced to describe not only entities (etc.) that must be supported, but it would also describe the constructs that should be exposed. It is in the exposed entity definitions that the subtype mandatory requirement would show itself since the section would state that the resource model entity could only be instantiated as its SUBTYPE.

This type of methodology eliminates any real or perceived interference when APs are brought together. If multiple APs have the same rule on an entities or combination of entities these become clear candidates for future AICs. In that process, the AP specific entities would become AIC specific entities.

The "problem" with this approach is that every AP will have many AP specific entities. This can cause more effort for implementors as other than AICs there would be fewer common leaf nodes in APs. This is a double edged sword. The other side to this is that the implementors are guaranteed a stable set of resources which can be implemented once. The implementation of APs would then be just working the AP specific entities and associated rules.

4 Other Stumbling Blocks

A suggestion has been made that the PDES, Inc. implementation methodology be refined to restrict the population of resource model entities where the data coming in is in an AP specific SUBTYPE.

This does not help. A case in point is AP 210. In this AP, **product** and **product_definition** are SUBTYPEd to death. Simply disregarding this data causes unacceptable data loss.

In section 3.2 of this document, the PDES, Inc. implementation methodology is discussed in reference to APs 202 and 203. The particular example was that one AP (203) included **product_definition_formation_with_specified_source** and the other (202) did not. While the Pdes, Inc. implementation methodology does "correct" this "problem", AP 203 should consider the removal of this entity as redundant data since the same information can be extrapolated in a more intelligent form from the supplied part relationship entities. The source indicator is a vestigial attribute left over from the PDES, Inc. CDIM-A1 pre-AP. This attribute was to be used to

indicate make or buy decisions and, in particular, was used to indicate this condition when the same company did both but the buy occurred internally. This can be done via supplied part relationships as long as the identification of the supplier is done with full description of address and organization relationship. The concept of an internal "buy" is only important to the company in question. This is really an application designation and probably should not be in the standard at all.

5 Summary

The information presented here is the author's view of the current situation in regard to APs sharing data (aka AP interoperability) in one or many repositories. This paper outlines some things that could be done to rectify some of the current sticking points or "problems". The author does not propose or assume that what is proposed here is the only possible solution to the indicated "problem".

In an effort to summarize, the PDES, Inc. proposed implementation methodology states that data sharing to the maximum extent possible was feasible if an AP implementation is given data from the implementation of another AP and the schema from that AP. It further states that when one AP attempts to read data defined in another AP it should not enforce its rules when accessing the data. The receiving AP rules are only invoked (or enforced) after the data is enhanced to meet the intent of the receiving AP and the receiving AP's schema name is added in an instance of **application_protocol_definition**.

The above methodology goes a long way to clearing up many of the real and perceived problems in the area of data sharing. The one remaining issue is what bearing the EXPRESS rule code in the AP schema has on the process. Some guidance must be provided on how to interpret these rules when APs are brought together. This paper suggests that anchoring these rules in the data is one option to correct this situation. The examples of this condition cited in this paper are but the tip of the iceberg. There are cardinality (requires), instantiation (subtype mandatory) and population (i.e. restrict) rules in most all APs. All of these types of rules must be dealt with in order to say that the AP interoperability (data sharing) "problem" is solved.